On Bottleneck Partitioning of k-ary n-cubes

David M. Nicol[*]
Weizhen Mao
The College of William and Mary
Williamsburg, VA 23185

Abstract

Graph partitioning is a topic of extensive interest, with applications to parallel processing. In this context graph nodes typically represent computation, and edges represent communication. One seeks to distribute the workload by partitioning the graph so that every processor has approximately the same workload, and the communication cost (measured as a function of edges exposed by the partition) is minimized. Measures of partition quality vary; in this paper we consider a processor's cost to be the sum of its computation and communication costs, and consider the cost of a partition to be the *bottleneck*, or maximal processor cost induced by the partition. For a general graph the problem of finding an optimal partitioning is intractable. In this paper we restrict our attention to the class of k-ary n-cube graphs with uniformly weighted nodes. Given mild restrictions on the node weight and number of processors, we identify partitions yielding the smallest bottleneck. We also demonstrate by example that some restrictions are necessary for the partitions we identify to be optimal. In particular, there exist cases where partitions that evenly partition nodes need not be optimal.

[*]This research was partially supported by the National Aeronautics and Space Administration under NASA contract number NAS1-19480 while the author was in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA, 23681.

1 Introduction

The problem of assigning workload in a parallel system has long been viewed as important, and in the general case, as intractable. A significant amount of research has addressed the problem of finding good, if not optimal, workload mappings; a number of different objective functions have been used. All relevant objective functions recognize that the quality of both load balance and communication costs are important. While workload imbalance is generally defined as a large deviation between the maximum and average load among processors, treatments of communication costs differ. A common technique is to measure the communication cost as the sum of all communication induced by the mapping. While this sometimes leads to more tractable treatments (e.g. [**?**, **?**]), it does not capture the fact that communication can happen in parallel. An alternative formulation is to assess the sum of computation and communication for each processor, and measure the quality of the mapping as the maximum processor load, or *bottleneck* [**?**, **?**]. The bottleneck measure does not take precedence relationships into consideration, and so is most useful in highly data-parallel computations where processors typically cycle through computation and communication phases.

In this paper we assume that a very regular graph—a k-ary n-cube[**?**]—describes the computation and communication needs of a data-parallel problem. Each node in the graph represents some piece of computational work, which we assume takes w time to perform. Each edge (i, j) represents some implicit communication necessary between nodes i and j; typically such an edge reflects a data dependency of node i's computation for the present iteration on the result of executing node j in the previous iteration (and vice-versa). The edges may be viewed as communication that must occur at the end of an iteration. We desire to partition the graph into p node sets, assigned one per processor, so as to minimize the bottleneck cost. The problem is not entirely academic. Several current parallel architectures have communication topologies based on the k-ary n-cube. The problem of partitioning a communication topology arises, for instance, when one executes a parallel simulation of traffic on a k-ary n-cube network [**?**, **?**].

The objective of this paper is to show that under mild restrictions on w and p, the optimal partition is intuitive, one that equi-partitions the graph into node sets that are internally clustered as tightly as possible. The main requirement turns out to be that p be large enough relative to the size of the k-ary n-cube. The central point of interest is that restrictions on w and p *are* needed; while intuitive, our results are not at all immediate. We also point out that previous analyses of partitioning regular grids differ from the current work in an subtle but important way. It is not the objective of the paper to give new partitioning algorithms, but to clarify one's intuition about partitioning k-ary n-cubes.

There are three bodies of work on graph partitioning that bear discussion. The technique of recursive spectral dissection (e.g., [**?**]) divides a graph into two pieces, based on an eigenvalue analysis of a matrix describing the graph connectivity. The algorithm is applied recursively until $p = 2^j$ node sets are defined. Each partition cut is guaranteed to achieve a certain level of load balance (not necessarily perfect balance), with a guaranteed upper bound on the number of edges cut. Spectral dissection may find some of the partitions we identify as optimal (when k is a power of two), but is not guaranteed to find them[1]. Recursive geometric partitioning (e.g. [**?**]) is similar in spirit, but different in details. A graph in $\mathcal{R}^n$ is projected onto the unit sphere in $\mathcal{R}^{n+1}$, and the projection is stretched to locate the center of mass (approximately) at the sphere's origin. A great

[1] Personal communication from Alex Pothen.

circle cut of the sphere partitions the node set into two pieces. The technique also guarantees a certain level of load balance and bounds the number of edges cut. Like spectral partitioning, the method *may* find the optimal partitions (in the same special case of k being a power of two), but also may not. On the other hand, recursive binary dissection [**?**] (and its extension, parametric recursive binary dissection [**?**]) will find the partitions we identify as optimal, when k is a power of 2. In the case of general graphs there is no such guarantee. The heuristic described in [**?**] is shown there to find optimal partitions of $N_{2,n}$, and obvious extensions to heuristic described in [**?**] will find *all* optimal partitions identified in this paper, provided the correct number of processors in each dimension are supplied in the problem description.

2 Problem Formulation

A k-ary n-cube $N_{k,n}$ is a graph with k^n nodes, with an edge defined between two nodes i and j if, in the base-k number system, the expressions of i and j differ in at most one digit, and differ there (modulo k) by exactly 1. Thus, if $i = b_{n-1}b_{n-2}\cdots b_0$ is the base-k representation, then in each dimension $j = 1, \ldots, n$, i shares an edge with $i' = b_{n-1}b_{n-2}\cdots(b_j + 1) \bmod k\ b_{j-1}\ \cdots b_0$ and with $i'' = b_{n-1}b_{n-2}\cdots(b_j - 1) \bmod k\ b_{j-1}\cdots b_0$. These edges are said to be in dimension j, and i' and i'' are said to be dimension j neighbors of i. Special cases include rings ($N_{k,1}$), hypercubes ($N_{2,n}$), two and three dimension toruses ($N_{k,2}$, $N_{k,3}$). It is useful to imagine $N_{k,n}$ as a collection of interconnected rings resident in an n-dimensional space.

A partition of $N_{k,n}$ into p subdomains is a collection of nonempty node subsets $\mathcal{P} = \{P_0, \ldots, P_{p-1}\}$. Abusing usual notation, we'll denote that an edge e has at least one endpoint in P_i by $e \in P_i$, and define the indicator function $I(e, P_i)$ to be one if exactly one of e's endpoints is in P_i, and zero otherwise. Then we denote the number of *external* edges in P_i by

$$Ext(P_i) = \sum_{e \in P_i} I(e, P_i),$$

denote the number of *internal edges* as

$$Int(P_i) = \sum_{e \in P_i} (1 - I(e, P_i)),$$

and define the *cost* of P_i as

$$C(P_i) = w|P_i| + Ext(P_i).$$

Here we weight the cost of each node by w to reflect the execution cost, where the communication cost associated with one edge is unity. The cost of $\mathcal{P}$ is taken as

$$B(\mathcal{P}) = \max_{0 \le i < p} C(P_i).$$

Given p and w, we wish to find the partition $\mathcal{P}$ that minimizes $B(\mathcal{P})$.

A very similar special case of this problem has been studied in the context of partitioning grids arising from the discretization of domains for the solution of partial differential equations, by Reed et al. [**?**]. It is instructive to consider the subtle difference in the problem specification, because the conclusions reached differ greatly.

The partitions considered by Reed et al. all tessellate a two-dimensional domain ($N_{k,2}$ without wraparound edges) with a common shape, e.g., rectangles, squares, or hexagons. The computation

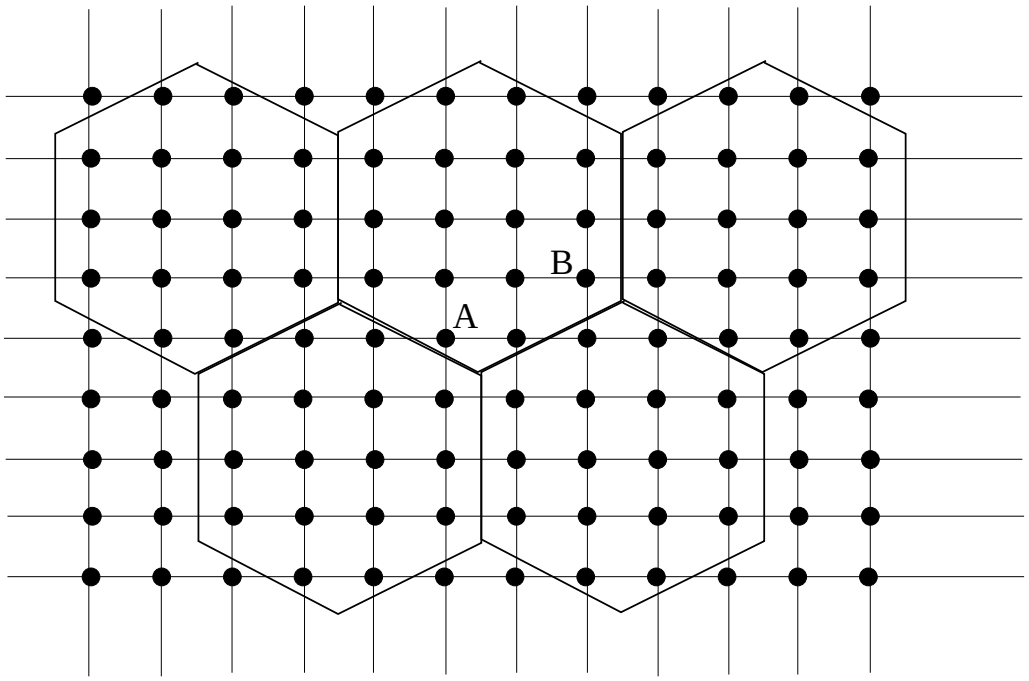


Figure 1: Hexagon partition in a 2d mesh

to communication ratio of different shapes are analyzed, but the communication cost is taken as the sum (over all grid points in the subgraph) of the cost of communicating each boundary point. This may vary from point to point. For instance, Figure **??** illustrates some hexes; point A has two edges cut, but since the endpoints of both edges are in the same hex, Reed et al. count the cost as one, not two. Point B has two edges cut, but both of these are counted. With this measure, the communication cost of a hex is taken as 10 although 14 edges are cut. Shapes like hexagons are shown to achieve a better computation/communication ratio than do squares. This is interesting, because in this case our results give general conditions under which squares are optimal, a significant difference due entirely to a minor change in the model of communication costs.

Reed et al.'s measure makes sense in its presented context where a specific numerical algorithm calls for the exchange of boundary value grid points. In other contexts unique edges from a node represent unique pieces of information, and the cost function we adapt is appropriate. We are aware of algorithms in computational fluid dynamics, for instance, where there is a unique "flow" along every edge in a mesh. Most of the grid partitioning community counts cut edges.

While our results identify general conditions under which equi-partitions are optimal for the bottleneck measure, it is worthwhile noting that this need not always be the case. An example that partitions a 6×6 mesh into 3 partition elements is shown in Figure **??**. Here the unbalanced partition has bottleneck cost $28w + 10$, the balanced partition has bottleneck cost $12w + 12$. The unbalanced partition is better whenever $w < 6/19$. This example illustrates the tension between partitioning to minimize computational imbalance and communication overhead. Our goal is find general conditions under which obvious equi-partitions are optimal with respect to the bottleneck metric.

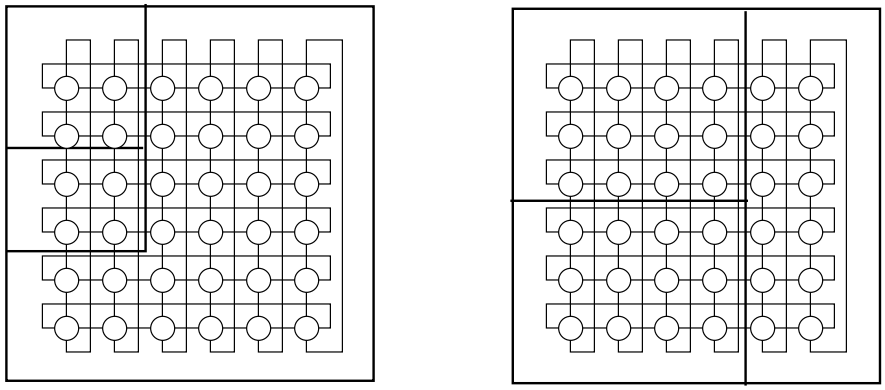


Figure 2: Equal sized partitions need not be optimal

3 Preliminaries

We first establish some preliminary results. These depend on k in a way that is captured by defining $T_k = 1$ for $k = 2$, and $T_k = 2$ for $k > 2$.

Observation 1 *Let A be any set of nodes in $N_{k,n}$. If $|A| = m$ and $Int(A) = v$, then $Ext(A) = T_k mn - 2v$.*

Lemma 2 *Let A be any set of nodes in $N_{k,n}$, $k \neq 3$, with $|A| = m$. Then $Int(A) \leq (m \log m)/2$. This bound is achieved when $m = 2^j$ for some $j \leq n$.*

Proof: We induct on m. The base case of $m = 1$ is trivially satisfied. Suppose then that the claim is true for any set of size $m - 1$ or smaller, and choose any node set A with $|A| = m$. Choose any two nodes x and y in A, consider their indices expressed in base-k notation and find a dimension j in which their indices differ in that notation. Let a and b be the dimension j index for x and y respectively. Viewing these indices as lying on a "ring" $0 - 1 - 2 - \cdots - (k-1) - 0$, cut the ring into two sequences of length 2 or greater, one of which contains a, and one of which contains b. Partition A into sets X_a and X_b, with X_a comprised of all nodes whose indices in dimension j lie in the same range as a's, and $X_b = A - X_a$. Let u and $m - u$ be the number of nodes in X and Y respectively. By the induction hypothesis, X_a has no more than $(u \log u)/2$ internal edges, and X_b has no more than $((m-u)\log(m-u)/2)$ internal edges. If $k = 2$ or if $k \geq 4$ there can be no more than $\min\{u, m-u\}$ edges between X_a and X_b, because any such edge has to connect nodes whose indices differ only in dimension j, and which must be adjacent on the ring we partitioned. Any node in either set can have at most one edge to the other set. It follows that A can have no more than

$$B_m(u) = (u \log u)/2 + ((m-u)\log(m-u))/2 + \min\{u, m-u\}.$$

Now the function

$$f_m(q) = (q \log q)/2 + ((m-q)\log(m-q))/2 + q$$

defined over $q \in [0, m/2]$ completely describes the bound as a function of $q = \min\{u, m-u\}$. Considered as a continuous function of q, analysis of derivatives reveals $f_m(q)$ to be convex over

$[0, m/2]$, and is hence maximized at the endpoint $q = m/2$. Simple algebra shows that $B_m(u) \leq f_m(m/2) = (m \log m)/2$, completing the induction. Finally, observe that the same argument holds in the case of $k = 2$ by relaxing the requirement that the dimension j ring be cut into lengths of 2 or greater—there is only one cut possible, and it is still possible for a node in X_a or X_b to have at most one edge between X_a and X_b. Finally, observe that when $m = 2^j$, $j \leq n$, the bound is achieved by any set A that forms a j-dimensional hypercube in $N_{k,n}$. ∎

Another bound is also useful. We will say that set A is *nowhere completed* if A contains no *completed rows*, i.e., no dimension j for which there are k nodes whose base-k indices all agree except in dimension j.

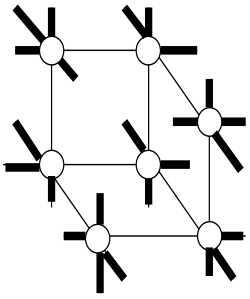
Lemma 3 *Let A be any set of nodes in $N_{k,n}$, $k > 2$, with $|A| = m$ such that A is nowhere completed. Then $Int(A) \leq n(m - m^{(n-1)/n})$. This bound is achieved whenever k is divisible by q, and $m = (k/q)^n$.*

Proof: By observation **??**, maximizing $Int(A)$ is equivalent to minimizing $Ext(A)$; we seek a set A' with m nodes minimizing $Ext(A')$. A' must be connected, otherwise we could always find a node set with smaller external edge count by translating a connected component linearly through $N_{k,n}$ until it eliminates one or more external edges by becoming adjacent to another connected component. Now represent the set as a "Manhattan polyhedron" (every face is parallel to some axis) formed by a collection of unit cubes in $\mathcal{R}^n$, each cube representing one node, and two cubes sharing a face if there is an edge between the nodes they represent. Figure **??** illustrates this construct. The number of external edges is thus equal to the number of exposed faces—the surface area of the Manhattan polyhedron. Now the surface area S_m of any Manhattan polyhedron in $\mathcal{R}^n$ is at least as large as that, say S_r, of the smallest "orthogonal polyhedron" (a rectangular solid in $\mathcal{R}^n$) that completely encloses it. Let $v \geq m$ be the volume of this orthogonal polyhedron. The polyhedron with volume v forming a perfect cube in $\mathcal{R}^n$ has surface area $S_c \leq S_r$. But the orthogonal polyhedron with volume m forming a perfect cube in $\mathcal{R}^n$ has smaller surface area yet. This minimal surface area is $2nm^{(n-1)/n} \leq Ext(A)$. The claimed bound on $Int(A)$ follows from observation **??**. Furthermore, whenever k is divisible by q, and $m = (k/q)^n$ we can construct a $(k/q) \times (k/q) \times \cdots (k/q)$ cube with exactly m nodes, in which case the bounds are exact. ∎

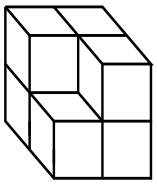
Our optimality results hold when the number of nodes in each partition set, $m = k^n/p$, is small enough to ensure that the optimal partition sets are nowhere completed. Since some internal edges are gained by forming a completed row (due to wrap-around), simple extensions to geometric arguments like those of Lemma **??** are not sophisticated enough to analyze these tradeoffs. However, a simple argument shows that for sets of size $m \leq k$, the configuration minimizing external edges need not have any completed rows.

Lemma 4 *For all $k \geq 2^n$ and $n \geq 2$ there exists a nowhere completed subset of k nodes in $N_{k,n}$ with minimal external edges.*

Proof: When $k \geq 2^n$ (and $n \geq 2$), the single configuration of k nodes that completes a row has exactly $2k(n-1)$ external edges, whereas the proof of Lemma **??** shows that the set of k nodes which is as cubelike as possible has no more than $2nk^{(n-1)/n}$ external edges. Now $2nk^{(n-1)/n} \leq 2k(n-1)$



Node set in 3d cube
External edges are highlighted



Manhattan polyhedron
Exposed faces represent external edges

Figure 3: Geometric interpretation of a connected node set

if and only if $(1/k) \leq (1-1/n)^n$. But $1/k \leq 0.25$ for all $k \geq 4$, and $(1-1/n)^n$ increases monotonically in n (converging to e^{-1}) and $(1-1/2)^2 = 0.25$. ∎

Proofs that optimally configured sets of size $m > k$ may be nowhere completed are beyond the scope of this note. However, we can put a lower bound on $Ext(A)$ for $|A| > k$, and analyze the relative error of this bound.

Lemma 5 *Let $k \geq 4$. For all $m \geq 2^n$ and $n \geq 2$, let $E_{m,n}$ be the minimal value of $Ext(A)$ among all node sets A with $|A| = m$. Then*

$$2nm^{(n-1)/n} - \frac{2m}{k} \leq E_{m,n} \leq 2nm^{(n-1)/n}.$$

Proof: The upper bound follows from the observation that among all sets A that are nowhere completed, $2nm^{(n-1)/n}$ is an upper bound on $Ext(A)$, and thus on $E_{m,n}$. The lower bound follows by subtracting from this the maximum number of external edges that may be deleted by completing a row—two per possible row. ∎

Now the relative difference between the upper and lower bound is $1 - m^{1/n}/(nk)$, which increases in m. Values of m we are most interested in derive from equi-partitions where every dimension is sliced identically. Let q divide k evenly, and let $m = (k/q)^n$. In this case the relative difference is $1 - 0.5/(qn)$. Consequently the bounds become tighter with increasing dimension size, n, and with decreasing partition size set $(k/q)^n$.

Let A be any set of nodes with $|A| = m$. From the observations above we see that

$$C_1(m) = wm + T_k mn - m \log m \leq C(A) \quad \text{for all } m = 1, 2, \cdots, k^n,$$

and

$$C_2(m) = wm + T_k mn - n(m - m^{(n-1)/n}) \leq C(A) \quad \text{for all } m = 1, 2, \cdots, k.$$

Observe that $C_2(m)$ is monotone non-decreasing, as $\frac{d}{dm} C_2(m) \geq 0$. Another result describes the relationship between C_1 and C_2.

Lemma 6 *For all $m \in [1, 2^n]$, $C_1(m) \geq C_2(m)$. For all $m > 2^n$, $C_1(m) \leq C_2(m)$.*

Proof: Analysis of derivatives with respect to m shows that $C_1'(1) \geq C_2'(1)$; since $C_1(1) = C_2(1)$ we infer that initially, for $x > 1$, $C_1(x) > C_2(x)$. Since both functions are continuous this dominance is maintained until the first m such that $C_1(m) = C_2(m)$. Algebra shows that the unique solution $m > 1$ is $m = 2^n$. At this point $C_1'(2^n) \leq C_2'(2^n)$, and the dominance reverses. ∎

4 Analysis of Cost Function

Since both $C_1(m)$ and $C_2(m)$ are lower bounds on $C(m)$, the function $C_3(m) = \max\{C_1(m), C_2(m)\}$ is a better composite bounding function. Previous observations have established that

$$C_3(m) = \begin{cases} C_1(m) & \text{for } m \leq 2^n \\ C_2(m) & \text{for } m \geq 2^n \end{cases} .$$

Furthermore, it is not difficult to show that $C_3(m)$ is concave over $m \in [1, 2^n]$, and that $C_3(m)$ is increasing over $m \in [2^n, k^n]$. Furthermore we also know that when $k \geq 4$, $C_3(m)$ is a lower bound on the cost of node set A with $|A| = m \leq k$ elements.

Our strategy now is to identify values of $m \leq k$ for which it is possible to partition $N_{k,n}$ into k^n/m isomorphic subgraphs, such that $C(m) = C_3(m)$. Since $C_3(m)$ is known to be increasing for $m > 2^n$, we determine conditions under which $C_3(m)$ is increasing over $[1, 2^n]$. Considered as a continuous function, the first derivative of $C_3(m)$ for $m \in [1, 2^n]$ is

$$\frac{d}{dm} C_1(m) = w + T_k n - \log m - 1/\ln 2.$$

This function decreases in m, and so will be non-negative over $[1, 2^n]$ if it is non-negative at $m = 2^n$. The latter condition is satisfied whenever $w + n(T_k - 1) \geq 1/\ln 2$. Thus

Lemma 7 *If $w > 1/\ln 2$ or if $k > 2$ and $n > 1$, then $C_3(m)$ is everywhere monotone non-decreasing over $[1, k^n]$.*

Monotonicity of $C_3(m)$ can be exploited, for if node sets $P_0, \ldots, P_{p-1}$ have sizes $m_0, \ldots, m_{p-1}$, then $\max\{C_3(m_0), \ldots, C_3(m_{p-1})\}$ is minimized when the node sets have equal sizes. To complete the analysis we simply identify conditions on p that ensure that $C_3(m) = C(P_i)$ for all $i = 0, \ldots, p-1$, and that $N_{k,n}$ can be partitioned into isomorphic node sets with this cost. Such partitions must be optimal.

Theorem 8 *The following are optimal partitions of $N_{k,n}$ with respect to the bottleneck cost.*

- *If some condition of Lemma* **??** *is satisfied, k is even, and $p = k^n/2^j$ with $j \leq n$, then $N_{k,n}$ may be partitioned into isomorphic hypercubes of dimension j.*
- *If some condition of Lemma* **??** *is satisfied, there is integer q such that $(k/q)^{1/n}$ is integer and $p = (k/q)^{(n-1)/n}$, then $N_{k,n}$ may be partitioned into isomorphic blocks of shape $(k/q)^{1/n} \times (k/q)^{1/n} \times \cdots \times (k/q)^{1/n}$.*

The partitions identified by this theorem are quite intuitive. They divide $N_{k,n}$ uniformly into equally sized sets of nodes, and the nodes in a set are clustered tightly. If the number of nodes in the set is less than 2^n, the nodes form a hypercube of some dimension no greater than n. If the number of nodes exceeds 2^n (but is no greater than k), they form a perfect cube in an n-dimensional space. However, while these optimal partitions are intuitive, we have already seen that perfectly balanced partitions need not be optimal. It is also noteworthy that the requirement on w for optimality disappears when p is small enough ($p \leq k^{(n-1)/n}$), or when $k > 2$.

A final result addresses the fact that restricting the number of nodes per processor to k or fewer may be overly conservative. For $k < m \leq (k/2)^n$ we can bound the deviation from optimal of cubic equi-partitions.

Lemma 9 *Let q divide k evenly, and consider the partitioning into adjacent blocks of size $(k/q) \times \cdots (k/q)$. Then the bottleneck cost is no more than $100/(nq)\%$ larger than optimal.*

Proof: Using $m = (k/q)$ Lemma **??** shows that the increase in external communication cost of the cubic partition is no more than $100/(nq)\%$. ∎

5 Conclusions

k-ary n-cubes are regular graph structures that are found in numerous contexts, especially in descriptions of communication networks. Partitioning of such graphs is a problem that arises in network design, and in parallelized simulation of such networks. This paper examines the problem of identifying optimal partitions of $N_{k,n}$ with respect to the bottleneck metric. Our investigations identify two points of interest. First, existing work on partitioning regular graphs for parallel processing has used a subtly different measure of communication, which leads to very different results than ours. Secondly, while the partitions we identify as optimal are intuitive, we show by example that equi-partitions need not always be optimal. Our results then help to delineate problems with intuitive optimal partitions from those with non-intuitive optimal partitions.

Open remaining problems that we are pursuing include dealing more conclusively with the effect of completing rows, and with determining the minimal value of w ensuring that equi-partitions are optimal.

References

[1] D. Agrawal, M. Choy, H.V. Leong, and A. Singh. Maya: A simulation platform for distributed shared memories. In *Proceedings of the 8^{th} Workshop on Parallel and Distributed Simulation*, pages 151–155, July 1994.

[2] A.Pothen, H.D. Simon, and K.P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal of Mathematical Analysis and Applications*, 11:430–452, 1990.

[3] M.J. Berger and S. H. Bokhari. A partitioning strategy for nonuniform problems on multiprocessors. *IEEE Trans. on Computers*, C-36(5):570–580, May 1987.

[4] S. Bokhari, T. Crockett, and D. Nicol. Parametric binary dissection. Technical Report 93-39, Institute for Computer Applications in Science and Engineering, 1993.

[5] S. H. Bokhari. Partitioning problems in parallel, pipelined, and distributed computing. *IEEE Trans. on Computers*, 37(1):48–57, January 1988.

[6] W. Dally. Performance analysis of k-ary n-cube interconnection networks. *IEEE Transactions on Computers*, 39(6):775–785, June 1990.

[7] P. Dickens, P. Heidelberger, and D. Nicol. A distributed memory lapse : Parallel simulation of message-passing programs. In *Proceedings of the 1994 Workshop on Parallel and Distributed Simulation*, pages 32–38, Edinburgh, Scotland, July 1994.

[8] B. Indurkhya, H. Stone, and L. Xi-Cheng. Optimal partitioning of randomly generated distributed programs. *IEEE Transactions on Software Engineering*, SE-12:483–495, March 1986.

[9] G. Miller, S.-H. Teng, W. Thurston, and S. A. Vavasis. Automatic mesh partitioning. In *Graph Theory and Sparse Matrix Computation*. Springer-Verlag, 1993.

[10] D. Nicol. Rectilinear partitioning of irregular data parallel computations. *Journal of Parallel and Distributed Computing*. To appear.

[11] D. Nicol and W. Mao. Automated parallelization of timed petri net simulations. Technical Report 93-91, Institute for Computer Applications in Science and Engineering, 1993.

[12] D.M. Nicol. Optimal partitioning of random programs across two processors. *IEEE Trans. on Software Engineering*, 15(2):134–141, 1989.

[13] D.M. Nicol and D.R. O'Hallaron. Improved algorithms for mapping parallel and pipelined computations. *IEEE Trans. on Computers*, 40(3):295–306, 1991.

[14] D. A. Reed, L. M. Adams, and M. L. Patrick. Stencils and problem partitionings: Their influence on the performance of multiple processor systems. *IEEE Trans. on Computers*, C-36(7):845–858, July 1987.